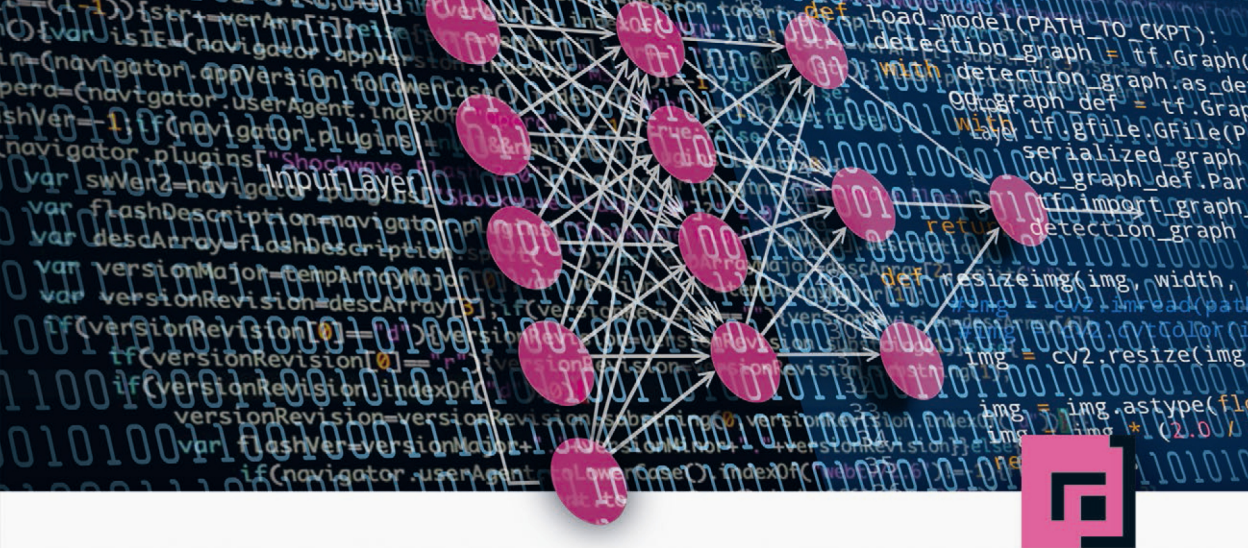




Paolo Perrotta

Machine Learning für Softwareentwickler

Von der Python-Codezeile zur
Deep-Learning-Anwendung

dpunkt.verlag



Zu diesem Buch – sowie zu vielen weiteren dpunkt.büchern – können Sie auch das entsprechende E-Book im PDF-Format herunterladen. Werden Sie dazu einfach Mitglied bei dpunkt.plus⁺:

www.dpunkt.plus

Paolo Perrotta

Machine Learning für Softwareentwickler

**Von der Python-Codezeile
zur Deep-Learning-Anwendung**



dpunkt.verlag

Paolo Perrotta

Lektorat: Dr. Michael Barabas

Übersetzung & Satz: G&U Language & Publishing Services GmbH, Flensburg, www.gundu.com

Copy-Editing: Claudia Lötschert, www.richtiger-text.de

Herstellung: Stefanie Weidner

Umschlaggestaltung: Helmut Kraus, www.exclam.de

Druck und Bindung: mediaprint solutions GmbH, 33100 Paderborn

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

ISBN:

Print 978-3-86490-787-6

PDF 978-3-96910-025-7

ePub 978-3-96910-026-4

mobi 978-3-96910-027-1

1. Auflage 2020

Translation Copyright für die deutschsprachige Ausgabe © 2020 dpunkt.verlag GmbH

Wiebinger Weg 17

69123 Heidelberg

Original Copyright © 2020 The Pragmatic Programmers, LLC.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

Original ISBN: 978-1-68050-660-0

Hinweis:

Dieses Buch wurde auf PEFC-zertifiziertem Papier aus nachhaltiger Waldwirtschaft gedruckt. Der Umwelt zuliebe verzichten wir zusätzlich auf die Einschweißfolie.



Schreiben Sie uns:

Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: hallo@dpunkt.de.

Die vorliegende Publikation ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Die Verwendung der Texte und Abbildungen, auch auszugsweise, ist ohne die schriftliche Zustimmung des Verlags urheberrechtswidrig und daher strafbar. Dies gilt insbesondere für die Vervielfältigung, Übersetzung oder die Verwendung in elektronischen Systemen.

Es wird darauf hingewiesen, dass die im Buch verwendeten Soft- und Hardware-Bezeichnungen sowie Markennamen und Produktbezeichnungen der jeweiligen Firmen im Allgemeinen warenzeichen-, marken- oder patentrechtlichem Schutz unterliegen.

Alle Angaben und Programme in diesem Buch wurden mit größter Sorgfalt kontrolliert. Weder Autor noch Verlag noch Übersetzer können jedoch für Schäden haftbar gemacht werden, die in Zusammenhang mit der Verwendung dieses Buches stehen.

Inhalt

Danksagung	xv
-------------------------	-----------

Wie um alles in der Welt ist so etwas möglich?	1
---	----------

Über dieses Buch	2
------------------------	---

Bevor wir beginnen	3
--------------------------	---

Teil 1 Von null auf Bilderkennung	7
---	----------

1 Einführung in Machine Learning	9
---	----------

Programmierung und Machine Learning im Vergleich	10
--	----

Überwachtes Lernen	12
--------------------------	----

Die Mathematik hinter dem Zaubertrick	14
---	----

Das System einrichten	18
-----------------------------	----

2 Ihr erstes ML-Programm	21
---------------------------------------	-----------

Die Aufgabenstellung	22
----------------------------	----

Pizzavorhersage mit überwachtem Lernen	22
--	----

Zusammenhänge in den Daten erkennen	23
---	----

Eine lineare Regression programmieren	25
---	----

Das Modell definieren	26
-----------------------------	----

Eine Vorhersage treffen	28
-------------------------------	----

Das Training implementieren	29
-----------------------------------	----

Los geht's!	33
-------------------	----

Bias hinzufügen	34
-----------------------	----

Zusammenfassung	37
-----------------------	----

Praktische Übung: Die Lernrate optimieren	39
---	----

3 Am Gradienten entlang	41
Unser Algorithmus bringt es nicht	42
Das Gradientenverfahren	43
Ein wenig Mathematik	45
Abwärts	46
Die dritte Dimension	48
Partielle Ableitung	49
Die Probe aufs Exempel	51
Probleme beim Gradientenverfahren	53
Zusammenfassung	55
Praktische Übung: Über das Ziel hinaus	56
 4 Hyperräume	 57
Noch mehr Dimensionen	58
Matrizenrechnung	61
Matrizen multiplizieren	61
Matrizen transponieren	65
Das ML-Programm erweitern	65
Die Daten aufbereiten	66
Die Vorhersagefunktion anpassen	68
Die Verlustfunktion anpassen	70
Die Gradientenfunktion anpassen	71
Der Code im Ganzen	71
Bye-bye, Bias!	73
Ein letzter Testlauf	74
Zusammenfassung	75
Praktische Übung: Statistik in der Praxis	75
 5 Ein binärer Klassifizierer	 77
Grenzen der linearen Regression	78
Invasion der Sigmoiden	81
Konfidenz	82
Glätten	84
Den Gradienten anpassen	86
Was ist mit der Modellfunktion geschehen?	87

Klassifizierung in Aktion	89
Zusammenfassung	91
Praktische Übung: Gewichtige Entscheidungen	92
6 Eine Aufgabe aus der Praxis	93
Die Daten	94
MNIST	94
Trainings- und Testdatensatz	96
Unsere eigene MNIST-Bibliothek	97
Die Eingabematrizen vorbereiten	97
Die Daten aufbereiten	100
Anwendung in der Praxis	102
Zusammenfassung	104
Praktische Übung: Knifflige Ziffern	104
7 Die große Herausforderung	105
Von zwei zu mehr Klassen	105
1-aus-n-Codierung	107
1-aus-n-Codierung in Aktion	109
Die Antworten des Klassifizierers decodieren	109
Mehr Gewichte	110
Die Matrixdimensionen überprüfen	111
Der Augenblick der Wahrheit	112
Zusammenfassung	115
Praktische Übung: Minensucher	115
8 Das Perzeptron	119
Gestatten, das Perzeptron!	119
Perzeptrone kombinieren	121
Die Grenzen von Perzeptronen	122
Linear separierbare Daten	123
Nicht linear separierbare Daten	125
Die Geschichte des Perzeptrons	126
Der entscheidende Schlag	127
Nachwehen	128

Teil 2 Neuronale Netze **129**

9 Das Netz entwerfen **131**

Ein neuronales Netz aus Perzeptronen zusammenstellen	132
Perzeptrone verketteten	133
Wie viele Knoten?	136
Die Softmax-Funktion	137
Der Entwurf.	139
Zusammenfassung	140
Praktische Übung: Auf eigene Faust	141

10 Das Netz erstellen **143**

Die Forward-Propagation programmieren	144
Die Softmax-Funktion schreiben	145
Die Klassifizierungsfunktionen schreiben	148
Kreuzentropie	148
Zusammenfassung	150
Praktische Übung: Test durch Zeitreise	151

11 Das Netz trainieren **153**

Wozu Backpropagation?	154
Von der Kettenregel zur Backpropagation	155
Die Kettenregel in einem einfachen Netz	155
Es wird komplizierter.	157
Backpropagation anwenden	160
Auf Kurs bleiben	161
Den Gradienten von w_2 berechnen	162
Den Gradienten von w_1 berechnen	163
Die Funktion back() erstellen	165
Die Gewichte initialisieren	166
Gefährliche Symmetrie	166
Tote Neuronen	167
Korrekte Gewichtsinitialisierung	169
Das fertige neuronale Netz	170
Zusammenfassung	173
Praktische Übung: Fehlstart	173

12 Funktionsweise von Klassifizierern	175
Eine Entscheidungsgrenze einzeichnen	175
Heimspiel für das Perzeptron	177
Klassifizierung verstehen	178
Eine Gerade reicht nicht aus	181
Die Entscheidungsgrenze krümmen	182
Zusammenfassung	184
Praktische Übung: Albtraumdaten	185
 13 Das Mini-Batch-Verfahren	 187
Der Lernvorgang grafisch dargestellt	188
Batch für Batch	190
Batches erstellen	191
Training mit Batches	193
Was geschieht bei verschiedenen Batchgrößen?	194
Ein Zickzackpfad	195
Große und kleine Batches	197
Vor- und Nachteile von Batches	198
Zusammenfassung	199
Praktische Übung: Das kleinste Batch	199
 14 Die Kunst des Testens	 201
Die Gefahr der Überanpassung	201
Das Problem mit dem Testdatensatz	204
Zusammenfassung	206
Praktische Übung: Überlegungen zum Testen	207
 15 Entwicklung	 209
Daten aufbereiten	210
Den Wertebereich der Eingabevariablen prüfen	211
Eingabevariablen standardisieren	211
Standardisierung in der Praxis	213
Die Hyperparameter anpassen	214
Die Anzahl der Epochen festlegen	215
Die Anzahl der verdeckten Knoten einstellen	215
Die Lernrate einstellen	218
Die Batchgröße festlegen	221

Der Abschlusstest	223
Auf dem Weg zu 99 %	225
Praktische Übung: 99 % erreichen	225
Zusammenfassung und Vorschau	226

Teil 3	Deep Learning	229
---------------	----------------------	------------

16	Tiefere Netze	231
-----------	--------------------------------	------------

Der Echidna-Datensatz	232
Neuronale Netze mit Keras erstellen	234
Den Aufbau des Netzes planen und der erste Code	235
Die Daten laden	237
Das Modell erstellen	237
Das Modell kompilieren	238
Das Netz trainieren	239
Die Entscheidungsgrenze einzeichnen	240
Das Netz ausführen	240
Ein tieferes Netz	242
Zusammenfassung	243
Praktische Übung: Keras-Spielwiese	244

17	Überanpassung vermeiden	247
-----------	--	------------

Was ist Überanpassung?	248
Ursachen der Überanpassung	248
Unteranpassung	252
Das Modell regularisieren	254
Eine Untersuchung unseres tiefen Netzes	255
L1- und L2-Regularisierung	258
Weitere Möglichkeiten zur Regularisierung	262
Zusammenfassung	264
Praktische Übung: Weitere Regularisierungstechniken	265

18	Tiefe Netze zählen	267
-----------	-------------------------------------	------------

Aktivierungsfunktionen	268
Wozu Aktivierungsfunktionen gut sind	269
Die sigmoide Aktivierungsfunktion und ihre Auswirkungen	270
Verschwindender Gradient	272

Alternativen zur Sigmoidfunktion	274
Gestatten, die ReLU-Aktivierungsfunktion	274
Die richtige Funktion auswählen	277
Weitere Techniken	279
Bessere Gewichtsinitialisierung	279
Gradientenabstieg auf Speed	280
Regularisierung für Fortgeschrittene	282
Batchnormalisierung	284
Zusammenfassung	285
Praktische Übung: Die 10-Epochen-Aufgabe	286
 19 Jenseits von Standardnetzen	289
Der CIFAR-10-Datensatz	290
Was ist CIFAR-10?	291
Das CIFAR-Waterloo	292
Die Bausteine von Faltungsnetzen	293
Ein Bild ist ein Bild	293
Faltung	294
Konvolutionsschichten	298
Ein Faltungsnetz ausführen	299
Zusammenfassung	302
Praktische Übung: Massenweise Hyperparameter	303
 20 Der Weg in die Tiefe	305
Der Aufstieg des Deep Learning	306
Es braut sich etwas zusammen	306
Der Wendepunkt	307
Fortsetzung folgt	308
Unverhältnismäßige Effektivität	309
Was nun?	312
Maschinelles Sehen	313
Sprache	313
Bildgenerierung	315
Das Gesamtbild	319
Praktischer Einstieg	320
Sie sind am Zug	321

Anhang	323
A Grundlagen von Python	323
Wie sieht Python-Code aus?	326
Dynamische Typisierung	327
Einrückungen	327
Die Bausteine von Python	328
Datentypen und Operatoren	329
Datenstrukturen	330
Strings	331
Schleifen	332
Funktionen definieren und aufrufen	333
Schlüsselwortargumente	334
Standardargumente	335
Module und Pakete	335
Module definieren und importieren	335
Das <code>__main__</code> -Idiom	337
Pakete verwalten	339
Objekte erstellen und verwenden	341
Das war's	342
 B Wörterbuch des Machine Learning	 345
Stichwortverzeichnis	367

Für meine Frau.

Na, wie klingt das, Irene?

Danksagung

Ein besonderes Dankeschön gilt meinen Fachgutachtern: Alessandro Bahgat, Arno Bastenhof, Roberto Bettazzoni, Guido »Zen« Bolognesi, Juan de Bravo, Simone Busoli, Pieter Buteneers, Andrea Cisternino, Sebastian Hennebrüder, Alberto Lumbreras, Russ Olsen, Luca Ongaro, Pierpaolo Pantone, Karol Przystalski, Dan Sheikh, Leonie Sieger, Gal Tsubery, l'ùmarèin pugnàtta di Casalecchio und Giancarlo Valente. Ich schulde euch eine ganze Menge! Ja, unter anderem auch ein Bier!

Vielen Dank auch den großzügigen Lesern, die mir in der Betaphase Kommentare und Errata geschickt haben: Marco Arena, Glen Aultman-Bettridge, Zbynek Bazanowski, Jamis Buck, Charles de Bueger, Leonardo Carotti, Helge Eichhorn, George Ellis, Bruno Girin, Elton Goci, Dave Halliday, Darren Hunt, Peter Lin, Karen Mauney, Bradley Mirly, Vasileios Ntarlagiannis, Volkmar Petschnig, David Pinto, Conlan Rios, Roman Romanchuk, Ionut Simion, Drew Thomas und Jeroen Wenting. Sollten wir uns mal auf einer Konferenz begegnen, dann sprecht mich ruhig an!

Danke auch an Annamaria Di Sebastiano, die mir die Geschichte vom Anfang des ersten Kapitels erzählt hat. Lange nicht mehr gesehen!

Des Weiteren danke ich Marc Schnierle für seine hervorragende Web-App `LaTeX4technics`¹, mit der ich die Formeln erstellt habe; Kimberly Geswein für die Gestaltung der Schriftart Indie Flower, die ich in den Diagrammen verwendet habe; den Designern der in der Originalausgabe ausgiebig verwendeten Schriftarten DejaVu Sans und DejaVu Sans Mono; und dem Stadtrat von Brisbane für die Veröffentlichung des Fotos mit dem Ameisenigel², das in leicht veränderter Form in einem der späteren Kapitel auftaucht.

Dieses Buch ging durch die Hände von drei hervorragenden Lektorinnen. Vielen Dank an Meghan Blanchette und Susan Conant, die mich von den ersten

1. <https://www.latex4technics.com>

2. <https://www.flickr.com/photos/brisbanecitycouncil/6971519658>

Entwürfen bis zur Mitte des Buchs brachten. Entschuldigt bitte, wenn ich hier noch Unterschiede mache, aber mein besonderer Dank gilt Katherine Dvorak für die unglaubliche Arbeit, die sie in dieses Buch investiert hat, und ihre feste, aber immer geduldige Führung. Du bist eine erstaunliche Lektorin, Katie.

Vielen Dank auch an Dad, Mom, Anna und Susanna sowie an meine erweiterte Familie von Freunden und Verwandten, die weit verstreut rund um die Welt leben. Es ist mir immer eine große Freude, euch um mich zu haben.

Und schließlich danke ich meiner Frau Irene: Du warst da, als ich mit dem Schreiben begann, und hast mich in diesem rauen Sommer sehr unterstützt. Jetzt, da das Buch in den Druck geht, bist du immer noch da, unerschütterlich und unbeirrbar. Ich verdanke dir alles. Dieses Buch ist dir gewidmet.

Wie um alles in der Welt ist so etwas möglich?

Machine Learning kann wie Zauberei wirken. Wie kann ein Computer die Objekte in einem Bild erkennen? Wie kann ein Auto selbstständig fahren?

Diese Leistungen sind nicht nur für Laien verblüffend, sondern auch für viele Softwareentwickler wie Sie und mich. Obwohl ich schon viele Jahre lang Code schrieb, war es mir völlig unklar, wie Machine Learning überhaupt funktionieren konnte. Während ich ein bisschen mit den neuesten Web-Frameworks herumbastelte, schrieben andere Leute faszinierende Software, die wie Science-Fiction wirkte und die ich einfach nicht fassen konnte.

Ich wollte mitmachen. Ich wollte selbst in der Lage sein, solche Dinge zu tun.

Da ich schon wusste, wie man Software schreibt, glaubte ich, ich könnte Machine Learning schnell begreifen. Wie schwer konnte das auch schon sein? Also setzte ich ein zuversichtliches Lächeln auf und widmete mich dem Studium dieses Themas. Mein zuversichtliches Lächeln gefror, als ich gegen ganze Batterien von Wänden anrannte.

Für Entwickler wie mich fühlt sich Machine Learning fremdartig an. Diese Disziplin quillt über von mathematischer Terminologie, akademischen Konventi-

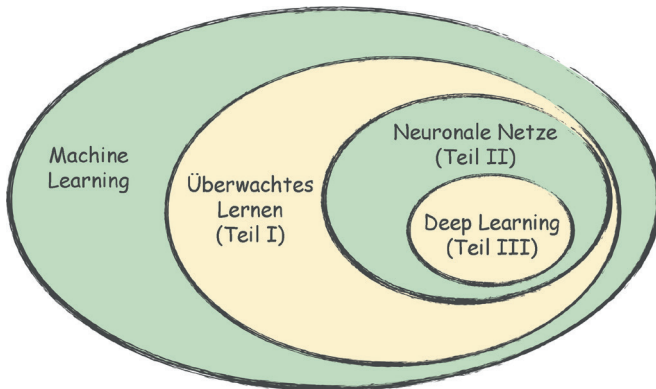
onen und, mal ganz ehrlich gesagt, schlechtem Code. Anstatt auf Tutorials werden Sie auf Vorträge und Forschungsberichte verwiesen. Für viele von uns ist Machine Learning ebenso einschüchternd wie faszinierend.

Dieses Buch ist das, was ich selbst gut hätte gebrauchen können, als ich mich mit Machine Learning zu befassen begann: eine Einführung, die sich an Entwickler richtet und in unserer Sprache geschrieben ist. Nach der Lektüre werden Sie mit den Grundlagen vertraut und in der Lage sein, Machine-Learning-Programme zu schreiben. Höchstwahrscheinlich werden Sie dann zwar immer noch nicht ihr eigenes selbstfahrendes Auto konstruieren können, aber zumindest werden Sie wissen, wie um alles in der Welt so etwas möglich ist.

Über dieses Buch

Dies ist ein Buch für Entwicklerinnen und Programmierer, die Machine Learning von der Pike auf lernen wollen.

Machine Learning ist ein sehr breit gefächertes Gebiet, das sich in einem einzelnen Buch nicht vollständig abhandeln lässt. Der Schwerpunkt liegt hier daher auf den drei Aspekten des Machine Learning, die heutzutage als die wichtigsten gelten: auf *überwachtem Lernen*, *neuronalen Netzen* und *Deep Learning*. Was es mit diesen Begriffen auf sich hat, werden wir uns im Verlauf dieses Buchs noch ansehen, aber für den Anfang finden Sie hier schon einmal eine Skizze sowie einige Definitionen in Kurzfassung:



- *Überwachtes Lernen* ist die heutzutage am weitesten verbreitete Spielart des Machine Learning. Teil I dieses Buchs, »Von null auf Bilderkennung«, gibt eine praktische Einführung in überwachtes Lernen. Schon im zweiten Kapitel werden wir ein minimales ML-Programm schreiben, das wir anschließend Schritt für Schritt erweitern und in ein als *Perzeptron* bezeichnetes Machine-Learning-System verwandeln. Unser Perzeptron ist dabei ein echtes Programm

für maschinelles Sehen, das leistungsfähig genug ist, um handgeschriebene Zeichen zu erkennen. Wir gestalten es komplett selbst, ohne dazu ML-Bibliotheken zu Hilfe zu nehmen, damit Sie jede einzelne Zeile des Codes genau verstehen.

- Es gibt viele verschiedene Möglichkeiten, um ein System für überwachtes Lernen zu implementieren. Am häufigsten werden dazu *neuronale Netze* verwendet. Dabei handelt es sich um einen brillanten Algorithmus, der sich an die Verknüpfung der Neuronen in unserem Gehirn anlehnt. Teil II dieses Buchs ist diesen Netzen gewidmet. Hier bauen wir das Programm aus Teil I zu einem richtiggehenden neuronalen Netz aus. Dazu müssen wir einige Probleme überwinden, aber die Mühe zahlt sich aus: Am Ende haben wir ein neuronales Netz, das weit leistungsfähiger ist als unser ursprüngliches Programm. Auch hier schreiben wir den Code wieder Zeile für Zeile selbst, sodass Sie selbst mit den internen Mechanismen experimentieren können.
- In den letzten Jahren wurden bei neuronalen Netzen erhebliche Fortschritte erzielt, als Forscher bahnbrechende Techniken zu ihrer Konstruktion und Verwendung erfanden. Diese moderne Technologie ist weit leistungsfähiger als die früheren einfachen neuronalen Netze, weshalb sie auch einen eigenen Namen erhalten hat: *Deep Learning*. Das ist auch der Titel von Teil III. Darin schreiben wir unser neuronales Netz mithilfe einer modernen ML-Bibliothek um. Der resultierende Code dient uns dann als Ausgangspunkt zur weiteren Erörterung von Deep Learning. Am Ende schauen wir uns noch einige anspruchsvolle Deep-Learning-Techniken an, womit wir auch die Grundlagen für Ihre weitere Beschäftigung mit diesem Thema legen.

Die Wirklichkeit ist natürlich nicht so sauber geordnet, wie unser Bild anzudeuten scheint. Beispielsweise kommen neuronale Netze nicht nur beim überwachten Lernen, sondern auch auf anderen Gebieten des Machine Learning zum Einsatz. Allerdings bildet dieses Diagramm einen guten Ausgangspunkt, um ein Gefühl für die einzelnen Themen in diesem Buch und ihre Beziehungen zueinander zu gewinnen.

Bevor wir beginnen

Dieses Buch macht Sie nicht über Nacht zu einem Machine-Learning-Profi, aber es kann Ihnen ein anschauliches, praktisches Verständnis dafür vermitteln, wie Machine Learning funktioniert. Ich will Ihnen einen Blick hinter die Kulissen dieser Disziplin gewähren, Ihnen die verborgenen Mechanismen zeigen und den Schleier des Geheimnisvollen lüften. Wenn Sie erst einmal die Grundprinzipien des Machine Learning kennen, ist es für Sie einfacher, selbst weiter nachzuforschen,

diese Techniken in Ihre Berufspraxis einfließen zu lassen und vielleicht sogar eine Karriere als ML-Ingenieur einzuschlagen.

Sie müssen kein erfahrener Entwickler sein, um dieses Buch verstehen zu können. Allerdings sollten Sie schon damit vertraut sein, kurze Programme zu schreiben. Wenn Sie Python kennen, dann umso besser, denn das ist die Sprache, die ich hier durchgängig verwende. Wenn nicht, ist das jedoch auch kein Beinbruch. Die Sprache ist sehr benutzerfreundlich, und der Code in diesem Buch ist nicht kompliziert. Zur Einführung können Sie Anhang A, »Grundlagen von Python«, lesen, und wenn Sie nicht mehr weiterwissen, finden Sie im Internet weitere Informationen.

Mit Machine Learning ist eine Menge Mathematik verbunden. Ich werde sie nicht vereinfachen, aber so anschaulich darstellen wie möglich. Für das Verständnis dieses Buchs brauchen Sie schon etwas Oberstufenmathematik. Ich gehe davon aus, dass Sie mit kartesischen Koordinatensystemen vertraut sind, wissen, was Achsen und ein Ursprung sind, und etwas mit einem Funktionsgraphen anfangen können. Darüber hinaus brauchen Sie nicht viel mathematisches Fachwissen. Sie können auch versuchen, das Buch zu lesen, wenn Sie glauben, ganz furchtbar schlecht in Mathe zu sein – aber bereiten Sie sich in diesem Fall schon einmal darauf vor, dass es vielleicht doch nicht möglich ist.

Wenn Sie sich dagegen mit linearer Algebra und Analysis gut auskennen, werden Ihnen einige der mathematischen Aspekte wahrscheinlich selbstverständlich erscheinen. In diesem Fall können Sie die Erklärungen, die Sie nicht brauchen, natürlich auch gern überspringen.

Mathematischer Hintergrund

Eine anschauliche Darstellung von mathematischen Dingen ist immer sehr hilfreich, aber manchmal sind auch formale Erklärungen sinnvoll. Wenn Sie bei der einen oder anderen Formel den Faden verlieren oder wenn Ihnen Mathematik liegt und Sie gern mehr über ein Thema erfahren wollen, dann schauen Sie sich die Kästen mit dem Titel »Mathematischer Hintergrund« an. Darin weise ich Sie auf die Mathematikektionen der hervorragenden Khan Academy¹ hin. Dort werden Sie etwas Passendes für sich finden, wie gut oder schlecht Ihre Vorkenntnisse in Mathematik auch immer sein mögen.

Um es ganz klar zu sagen: Diese zusätzlichen Lektionen sind optional. Es ist nicht erforderlich, sie sich anzusehen, um dieses Buch verstehen zu können. Sie dienen lediglich dazu, Ihre Kenntnisse über die Mathematik des Machine Learning zu vertiefen.

1. www.khanacademy.org

Machine Learning bringt seine eigene umfangreiche und besondere Terminologie mit. Sie werden daher wahrscheinlich auf neue Wörter und auf neue Bedeutungen bekannter Wörter stoßen. Gehen Sie es ruhig an. Sie müssen sich nicht sofort alles merken. Im Zweifelsfall können Sie immer in Anhang B, »*Wörterbuch des Machine Learning*«, nachschlagen.

Viele der Datenmengen, die ich als Beispiele verwende, bestehen aus Bildern. Mit Machine Learning können Sie jedoch mehr tun, als nur Bilder zu erkennen: Sie können Text analysieren, Musik generieren und sogar Unterhaltungen führen. Allerdings eignet sich gerade die Bilderkennung für anschauliche Beispiele, weshalb ich diese Disziplin im ganzen Buch als Standardanwendung benutze.

Es gibt auch einige Onlinere Ressourcen, die Sie kennen sollten. Eine davon ist die offizielle Webseite² zur Originalausgabe dieses Buchs auf Pragmatic Bookshelf. Von dort können Sie auch den Quellcode der Beispiele herunterladen.

Des Weiteren gibt es zu diesem Buch eine Begleitwebsite namens ProgML³. Sie enthält (in englischer Sprache) einige zusätzliche Erklärungen und Einzelheiten, die nicht mehr in dieses Buch passten. An den entsprechenden Stellen im Text habe ich Verweise darauf angegeben.

Genug der Vorrede – legen wir nun los mit Teil I!

Verweise auf ProgML
sind auf diese Weise
gestaltet.

2. www.dpunkt.de/mlprog

3. www.progml.com

Teil 1

Von null auf Bilderkennung

Dieser Teil gibt eine Einführung in überwachtes Lernen. Innerhalb von nur zwei Kapiteln werden wir ein erstes Machine-Learning-System programmieren, das wir anschließend Schritt für Schritt weiterentwickeln, bis es leistungsfähig genug ist, um handgeschriebene Ziffern zu erkennen.

Ja, Sie haben richtig gelesen: Auf den nächsten *etwa 100 Seiten* werden wir ein Programm zur Bilderkennung schreiben. Was noch besser ist: Wir werden dazu keine Machine-Learning-Bibliothek verwenden. Abgesehen von einigen Allzweckfunktionen für arithmetische Berechnungen und grafische Darstellung schreiben wir den ganzen Code selbst.

Es ist sehr unwahrscheinlich, dass Sie in Ihrer zukünftigen Karriere jemals Machine-Learning-Algorithmen von Grund auf selbst schreiben müssen. Aber es einmal zu tun, um die Grundlagen richtig zu verstehen, ist von unschätzbarem Wert. Sie werden genau wissen, was jede einzelne Zeile des fertigen Programms tut. Danach wird Machine Learning für Sie nie wieder wie schwarze Magie wirken.

1

Einführung in Machine Learning

Softwareentwickler erzählen sich gern Veteranengeschichten. Sobald ein paar von uns in einer Kneipe zusammensitzen, fragt einer: »An was für Projekten arbeitet ihr gerade?« Dann nicken wir heftig und hören uns die amüsanten und teilweise furchtbaren Geschichten der anderen an.

Bei einem dieser abendlichen Geplänkel Mitte der 90er erzählte mir eine Freundin von dem unmöglichen Auftrag, an dem sie gerade arbeitete. Ihre Vorgesetzten wünschten sich von ihr ein Programm, das Röntgenaufnahmen analysieren und dadurch Krankheiten erkennen konnte, etwa eine Lungenentzündung.

Meine Freundin warnte die Geschäftsleitung vor, dass das ein hoffnungsloses Unterfangen sei, aber man wollte ihr nicht glauben. Wenn ein Radiologe das leisten konnte, so argumentierten die Manager, warum dann nicht auch ein Visual-Basic-Programm? Sie stellten ihr sogar einen Radiologen zur Seite, damit sie lernte, wie er vorging, und dies in Code umsetzen konnte. Diese Erfahrung bestärkte sie jedoch nur in ihrer Meinung, dass Radiologie menschliches Urteilsvermögen und menschliche Intelligenz erforderte.

Wir lachten über die Sinnlosigkeit dieser Aufgabe. Ein paar Monate später wurde das Projekt aufgegeben.

Doch kehren wir nun in die Gegenwart zurück. 2017 veröffentlichte ein Forschungsteam der Stanford University einen Algorithmus, um Lungenentzündung anhand von Röntgenbildern zu erkennen.¹ Er erfüllte nicht nur seine Aufgabe, sondern war sogar zuverlässiger als ein professioneller Radiologe. Das hatte als unmöglich gegolten. Wie hatten die Forscher es geschafft, diesen Code zu schreiben?

Die Antwort lautet: gar nicht. Anstatt Code zu schreiben, setzten sie Machine Learning ein. Sehen wir uns an, was das bedeutet.

Programmierung und Machine Learning im Vergleich

Das folgende Beispiel zeigt den Unterschied zwischen Machine Learning (oder kurz ML) und gewöhnlicher Programmierung. Stellen Sie sich vor, Sie sollen ein Programm erstellen, das Videospiele spielt. Bei der traditionellen Programmierung würden Sie dazu Code wie den folgenden schreiben:

```
enemy = get_nearest_enemy()
if enemy.distance() < 100:
    decelerate()
    if enemy.is_shooting():
        raise_shield()
    else:
        if health() > 0.25:
            shoot()
        else:
            rotate_away_from(enemy)
else:
    # ... und noch viel mehr Code
```

Und so weiter. Der Großteil des Codes würde aus einer Riesenmenge von `if...else`-Anweisungen vermischt mit Befehlen wie `shoot()` bestehen.

Moderne Sprachen bieten uns zwar Möglichkeiten, diese hässlichen, verschachtelten `if`-Anweisungen durch angenehmere Konstruktionen wie Polymorphismus, Mustererkennung oder ereignisgestützte Aufrufe zu ersetzen, aber das Grundprinzip der Programmierung bleibt unverändert: Sie sagen dem Computer, wonach er Ausschau halten und was er tun soll. Dabei müssen Sie jede mögliche Bedingung aufführen und jede mögliche Aktion definieren.

Mit dieser Vorgehensweise sind wir weit gekommen, aber sie hat auch einige Nachteile. Erstens dürfen Sie nichts auslassen. Wahrscheinlich können Sie sich Dutzende oder gar Hunderte von besonderen Situationen vorstellen, die Sie in dem Videospielprogramm berücksichtigen müssen. Was geschieht, wenn sich ein Gegner nähert, sich aber ein Power-up zwischen Ihnen und ihm befindet, das Sie vor

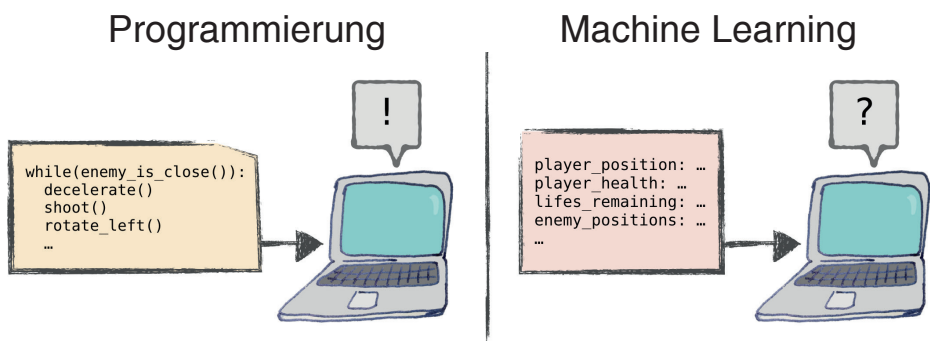
1. news.stanford.edu/2017/11/15/algorithm-outperforms-radiologists-diagnosing-pneumonia

seinen Schüssen schützen kann? Ein menschlicher Spieler wird eine solche Situation schnell erkennen und zu seinem Vorteil nutzen. Kann ein Programm das auch? Das hängt ganz von dem Programm ab. Wenn Sie diesen Sonderfall beim Schreiben des Codes berücksichtigt haben, dann kann das Programm damit umgehen. Allerdings wissen wir, wie schwer es ist, selbst in so eng umrissenen Aufgabenfeldern wie der Buchhaltung jegliche Sonderfälle abzudecken. Wenn Sie sämtliche Sonderfälle auf so komplexen Gebieten wie dem Spielen von Videospielen, dem Fahren eines Lkw oder dem Erkennen eines Bilds auflisten wollen, kann ich Ihnen dazu nur viel Glück wünschen.

Es reicht aber nicht nur, all diese Fälle aufzulisten; Sie müssen auch wissen, wie Sie dabei jeweils eine Entscheidung fällen. Das ist die zweite große Einschränkung bei der Programmierung, die in manchen Fachgebieten schon das Aus bedeutet. Betrachten Sie zum Beispiel eine Aufgabe aus dem Bereich des *maschinellen Sehens* wie das bereits erwähnte Erkennen einer Lungenentzündung anhand einer Röntgenaufnahme.

Wir wissen nicht *genau*, wie ein Radiologe eine Lungenentzündung erkennt. Wir haben zwar eine grobe Vorstellung davon, dass er nach undurchsichtigen Bereichen sucht, aber wir wissen nicht, wie sein Gehirn solche undurchsichtigen Bereiche erkennt und bewertet. Manchmal kann ein solcher Experte selbst nicht erklären, wie er zu der Diagnose gekommen ist, sondern nur vage Begründungen geben wie: »Ich weiß aus Erfahrung, dass eine Lungenentzündung nicht so aussieht.« Da wir nicht wissen, wie solche Entscheidungen ablaufen, können wir einen Computer auch nicht anweisen, sie zu fällen. Dieses Problem stellt sich bei allen typisch menschlichen Aufgaben, etwa dem Verkosten von Bier oder dem Verstehen eines Satzes.

Machine Learning dagegen stellt das Prinzip der traditionellen Programmierung auf den Kopf: Der Computer erhält keine *Anweisungen*, sondern *Daten*, und wird aufgefordert, selbst herauszufinden, was er tun soll.



Dass der Computer selbst etwas herausfinden soll, klingt wie Wunschdenken. Allerdings gibt es tatsächlich einige Möglichkeiten, um das zu erreichen. Für alle diese Vorgehensweisen ist nach wie vor die Ausführung von Code erforderlich, allerdings ist dieser Code im Gegensatz zur herkömmlichen Programmierung keine schrittweise Anleitung, um das vorliegende Problem zu lösen. Beim Machine Learning teilt der Code dem Computer mit, wie er die Daten verarbeiten soll, um das Problem selbst zu lösen.

Betrachten wir als Beispiel wieder einen Computer, der herausfinden soll, wie man ein Videospiel spielt. Stellen Sie sich einen Algorithmus vor, der durch Versuch und Irrtum zu spielen lernt. Zu Anfang gibt er zufällige Befehle: schießen, bremsen, drehen usw. Er merkt sich, wann die Befehle zum Erfolg führen, etwa zu einem höheren Punktestand, aber auch, wann sie in einem Fehlschlag wie dem Tod der Spielfigur resultieren. Außerdem registriert er den Spielzustand: wo sich die Gegner, die Hindernisse und die Power-ups befinden, wie viele Gesundheitspunkte die Figur hat usw.

Wenn der Algorithmus später einem ähnlichen Spielzustand begegnet, wird er mit höherer Wahrscheinlichkeit erfolgreiche Aktionen durchführen. Eine Vielzahl solcher Trial-and-Error-Durchläufe kann aus dem Programm einen fähigen Spieler machen. Mit dieser Vorgehensweise erreichte ein System im Jahr 2013 geradezu übermenschliche Fähigkeiten bei einer Reihe alter Atari-Spiele.²

Diese Form von ML wird als *bestärkendes Lernen* oder *Reinforcement Learning* bezeichnet. Es funktioniert fast genauso wie Hundetraining: »Gutes« Verhalten wird belohnt, sodass der Hund es häufiger an den Tag legt. (Ich habe das auch mit meiner Katze versucht, aber bislang ohne Erfolg.)

Reinforcement Learning ist jedoch nur eine Möglichkeit, mit der ein Computer die Lösung eines Problems herausfinden kann. Der Schwerpunkt dieses Buchs liegt dagegen auf einer anderen Form von Machine Learning, nämlich der wahrscheinlich am häufigsten verwendeten. Sehen wir uns das genauer an.

Überwachtes Lernen

Unter den verschiedenen Ansätzen des ML hat das *überwachte Lernen* (*Supervised Learning*) bisher die beeindruckendsten Ergebnisse hervorgebracht. Damit lassen sich auch Aufgaben wie die Diagnose einer Lungenentzündung lösen.

Ausgangspunkt des überwachten Lernens ist eine Reihe von *Beispielen*, die jeweils mit *Labels* versehen sind, aus denen der Computer etwas lernen kann:

2. deepmind.com/research/publications/playing-atari-deep-reinforcement-learning

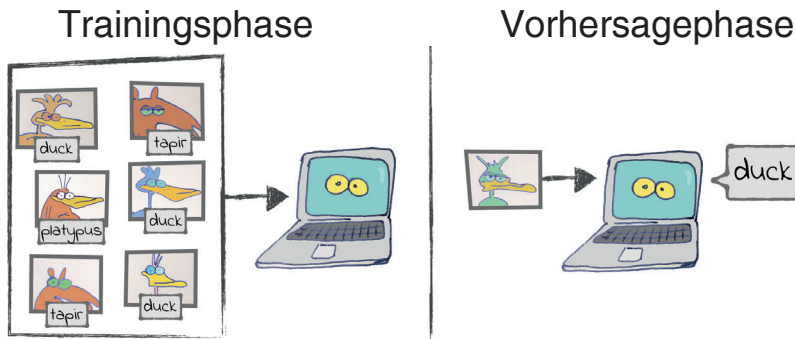
Zweck	Beispiel	Label
Ein System, das aus dem Bellen eines Hundes die Hunderasse bestimmt	.wav-Datei mit Hundegbell	Die Hunderasse, also etwa »Windhund«, »Beagle« usw.
Ein System, das eine Lungenentzündung erkennen kann	Röntgenaufnahme	Ein boolesches Flag: 1, wenn die Aufnahme eine Lungenentzündung zeigt, anderenfalls 0
Ein System, das die Einnahmen einer Eisdiele anhand des Wetters vorhersagt	Aufgezeichnete Temperaturen an einem vergangenen Tag	Die aufgezeichneten Einnahmen an diesem Tag
Ein System, das erkennt, welche Stimmung der Tweet eines Politikers vermittelt	Ein Tweet	Eine Stimmung wie »entsetzt«, »verärgert« oder »wütend«

Wie Sie sehen, können als Beispiele viele verschiedene Dinge genutzt werden: Daten, Text, Klänge, Videos usw. Es gibt auch zwei Arten von Labels: *Numerische Labels* sind einfach nur Zahlen, wie es etwa bei dem Temperatur-Eisdielenumsätze-Umrechner der Fall ist. *Kategorielabels* dagegen stehen wie bei dem Hunderassendetektor für eine Kategorie (oder Klasse) aus einer vordefinierten Menge. Wenn Sie Ihre Fantasie spielen lassen, werden Ihnen noch viele weitere Möglichkeiten einfallen, um etwas – seien es Zahlen oder Kategorien – aus etwas anderem zu bestimmen.

Nehmen wir an, dass wir bereits eine Reihe solcher Beispiele mit Labels zusammengestellt haben. Daran schließen sich nun die beiden folgenden Phasen des überwachten Lernens an:

1. In der ersten Phase speisen wir die mit Labeln gekennzeichneten Beispiele in einen Algorithmus ein, der Muster erkennen kann. Ein solcher Algorithmus kann beispielsweise bemerken, dass alle Aufnahmen, die eine Lungenentzündung zeigen, gemeinsame Merkmale aufweisen, etwa undurchsichtige Bereiche, die in den Aufnahmen ohne Lungenentzündung nicht zu finden sind. Dies ist die sogenannte *Trainingsphase*, in der der Algorithmus die Beispiele immer wieder durchsieht, um zu lernen, solche Muster zu erkennen.
2. Wenn der Algorithmus weiß, wie eine Lungenentzündung aussieht, gehen wir zur *Vorhersagephase* über, in der wir die Früchte unserer Arbeit ernten. Wir zeigen dem trainierten Algorithmus eine Röntgenaufnahme *ohne Label*, woraufhin uns der Algorithmus sagt, ob sie Anzeichen einer Lungenentzündung aufweist oder nicht.

Auch das folgende System zur Erkennung von Tierarten ist ein Beispiel für überwachtes Lernen. Hier sind die einzelnen Eingaben die Bilder von Tieren mit der Angabe der Art als Label. In der Trainingsphase zeigen wir dem Algorithmus Bilder mit Labels und in der Vorhersagephase Bilder ohne Labels, wobei der Algorithmus die Labels rät:



Ich habe bereits gesagt, dass der Computer beim Machine Learning etwas aus den Daten »herausfindet«. Das überwachte Lernen ist ein Beispiel eines solchen Vorgangs. Bei der herkömmlichen Programmierung führen Sie den Computer mit Ihrem Code von der Eingabe zur Ausgabe. Beim überwachten Lernen dagegen stellen Sie dem Computer Beispiele von Ein- und Ausgaben zur Verfügung, anhand derer er selbst herausfindet, wie er von den einen zu den anderen kommt.

Möglicherweise hat dieser grobe Überblick über das überwachte Lernen mehr Fragen aufgeworfen als beantwortet. Wir haben gesagt, dass ein Programm zum überwachten Lernen Muster und gemeinsame Merkmale in den Daten »erkennt«. Aber wie macht es das? Schauen wir uns also genauer an, wie dieses Zauberstück tatsächlich funktioniert.

Die Mathematik hinter dem Zaubertrick

Um die Beziehung zwischen Daten und ihren Labels zu erkennen, nutzt ein System zum überwachten Lernen das mathematische Prinzip der *Näherungsfunktion* aus. Sehen wir uns an einem konkreten Beispiel an, was das bedeutet.

Nehmen wir an, Sie haben eine Solaranlage auf Ihrem Dach und wollen ein System zum überwachten Lernen konstruieren, das lernt, in welchem Maße diese Anlage Energie erzeugt, und die in Zukunft generierten Energiemengen vorhersagen kann.

Der Ertrag einer Solaranlage hängt von verschiedenen Variablen ab, darunter der Tageszeit, dem Wetter usw. Besonders wichtig scheint die Tageszeit zu sein, weshalb Sie sich darauf konzentrieren. In der typischen Vorgehensweise für über-